

Génie logiciel avancé

Preuve de programmes

Delphine Longuet
delphine.longuet@lri.fr

<http://www.lri.fr/~longuet/Enseignements/16-17/L3-GLA>

Avantages et limitations du test

Avantages :

- Exécution du système
- Environnement et conditions réels d'utilisation
- Facilité de mise en œuvre
- Écriture possible des tests avant le développement
- Validation permanente des changements au sein du code

Principal inconvénient : incomplétude

Impossible d'exécuter des tests pour toutes les données d'entrée possibles d'un programme. Pas de certitude sur l'absence de fautes, seulement une certaine confiance dans la qualité

Vérification de programmes

But : démontrer mathématiquement l'absence de fautes dans un programme

Méthodes :

- **Analyse statique** (model-checking, interprétation abstraite) : démonstration de l'absence de blocage, de débordement d'entier, de débordement de tableau...
- **Raffinement** : code construit par raffinements successifs d'un modèle formel, code correct par construction
- **Vérification déductive** : à l'aide d'un modèle formel du langage de programmation, démonstration que le programme satisfait sa spécification

Utilisation dans l'industrie

Ligne 14 du métro à Paris : logiciel critique embarqué développé selon la méthode B (1998, approche par raffinement)

Logiciel de contrôle de vol de l'A380 : vérification de l'absence d'erreur à l'exécution avec Astree (2005, interprétation abstraite)

Hyperviseur de Microsoft : correction prouvée avec VCC et le prouveur automatique Z3 (2008, vérification déductive)

Plus récemment : vérification de PikeOS

Compilateur C certifié : développé avec l'assistant de preuve Coq (2009, code correct par construction généré par un assistant de preuve)

Vérification déductive

Ingrédients nécessaires :

- **Spécification formelle du programme** : pré et post-conditions
- **Modèle formel du langage de programmation** : signification mathématique de chacune des expressions et instructions de ce langage

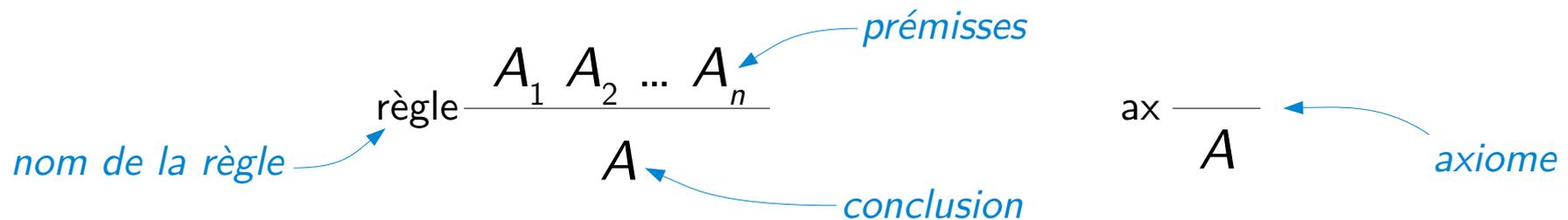
Résultat obtenu : **démonstration mathématique** que le programme satisfait sa spécification, c'est-à-dire

Pour toutes les entrées du programme vérifiant la pré-condition, si le programme termine, la post-condition est vérifiée

Rappels : système d'inférence

Système d'inférence : ensemble de règles permettant de déduire des faits depuis des axiomes et des hypothèses

Axiome : règle sans prémisse



Si A_1, A_2, \dots, A_n sont démontrées, alors la règle permet de démontrer A

Exemple : système d'inférence pour l'égalité

$$\begin{array}{l} \text{ref} \frac{}{x = x} \quad \text{sym} \frac{x = y}{y = x} \quad \text{trans} \frac{x = y \quad y = z}{x = z} \quad \text{subs} \frac{x = y \quad P(x)}{P(y)} \end{array}$$

Rappels : arbre de preuve

Arbre de preuve : séquence d'applications des règles d'un système inférence permettant de démontrer un fait

$$\begin{array}{c}
 \text{sym} \frac{f(a,b) = a}{a = f(a,b)} \quad \text{subs} \frac{f(a,b) = a \quad f(f(a,b),b) = c}{f(a,b) = c} \\
 \text{trans} \frac{a = f(a,b) \quad f(a,b) = c}{a = c} \quad \text{ref} \frac{}{g(a) = g(a)} \\
 \text{subs} \frac{a = c \quad g(a) = g(a)}{g(a) = g(c)}
 \end{array}$$

hypothèses

Conclusion dérivable à partir des hypothèses, noté :

$$\{f(a,b) = a, f(f(a,b),b) = c\} \vdash g(a) = g(c)$$

Rappels : théorème

Théorème : fait dérivable à partir d'hypothèses par l'application des règles d'un système d'inférence

Conclusion d'un arbre de preuve dont les feuilles sont des axiomes et des hypothèses

$$\{ \text{Hyp}_1, \dots, \text{Hyp}_n \} \vdash \text{Th}$$

$$\frac{\frac{\text{Hyp}_2 \quad \dots}{\text{---}} \text{ax} \quad \frac{\dots \quad \text{Hyp}_n}{\text{---}} \text{ax}}{\dots \quad \dots} \text{---}}{\text{Hyp}_1 \quad \dots}{\text{---}} \text{Th}$$

Langage impératif simple

Types : booléens et entiers

Instructions :

SKIP	aucun effet
$x := \text{exp}$	affectation
$\text{ins}_1 ; \text{ins}_2$	séquence d'instructions
IF cond THEN ins_1 ELSE ins_2	branchement conditionnel
WHILE cond DO ins	boucle

avec :

x : variable entière ou booléenne

exp : expression arithmétique ou booléenne

cond : expression booléenne

$\text{ins}, \text{ins}_1, \text{ins}_2$: instructions

Spécification d'un programme

Programme *prog* : séquence d'instructions

Pré-condition *P* : propriété à vérifier pour exécuter le programme

Post-condition *Q* : propriété à vérifier après l'exécution du programme

Triplet de Hoare : $\{P\} prog \{Q\}$

Signification : si le programme *prog* est exécuté sur des données d'entrée satisfaisant la pré-condition *P*, alors si l'exécution termine, la post-condition *Q* est vérifiée



Correction partielle : aucune information si le programme ne termine pas

Triplet de Hoare

Exemples de triplets de Hoare valides :

$$\{x = 1\} x := x+2 \{x = 3\}$$

$$\{y = 2\} x := y+5 \{x = 7\}$$

$$\{y = 1\} x := y+1 ; z := x-1 \{z = 1\}$$

$$\{-4 \leq x \leq 4\} \text{ IF } x \geq 0 \text{ THEN } x := x-2 \text{ ELSE } x := x+2 \{-2 \leq x \leq 2\}$$

$$\{x \geq 0\} \text{ WHILE } x > 0 \text{ DO } x := x-1 \{x = 0\}$$

Triplet de Hoare

Triplets de Hoare particuliers :

$\{true\} \text{ prog } \{Q\}$: le programme peut toujours être exécuté,
aucune condition sur les entrées

$$\{true\} \ x \ := \ 2 * x \ \{x \bmod 2 = 0\}$$

$\{false\} \text{ prog } \{Q\}$: le programme ne peut jamais être exécuté,
aucune valeur d'entrée n'est possible. Le triplet est valide pour tout Q

$$\{false\} \ x \ := \ 25 \ \{x = 0\}$$

$\{P\} \text{ prog } \{false\}$: le programme ne termine pas, aucune valeur de
sortie n'est possible. Le triplet est valide pour tout P

$$\{b = true\} \text{ WHILE } b \text{ DO SKIP } \{false\}$$

Calcul de Hoare

Systeme d'inférence pour le langage impératif simple présenté

Une règle pour chaque type d'instruction

- axiome pour SKIP
- axiome pour l'affectation
- règle pour la séquence
- règle pour le branchement conditionnel
- règle pour la boucle

+ règle de conséquence : règle logique sur les pré et post-conditions

Calcul de Hoare : SKIP

Axiome pour SKIP

$$\frac{}{\{P\} \text{ SKIP } \{P\}} \text{ skip}$$

SKIP n'a aucun effet, donc toute propriété vraie avant est vraie après

Calcul de Hoare : affectation

Axiome pour l'affectation

$$\frac{}{\{P[x \mapsto exp]\} x := exp \{P\}} \text{aff}$$

où $P[x \mapsto exp]$ est la formule P dans laquelle toutes les occurrences de x ont été remplacées (syntaxiquement) par exp (substitution)

Pour que P soit vraie après l'affectation, il suffit que $P[x \mapsto exp]$ soit vraie avant



À appliquer de droite à gauche : ~~$\{x = y\} x := 5 \{??\}$~~

Calcul de Hoare : affectation

Axiome pour l'affectation

$$\frac{}{\{P[x \mapsto \text{exp}]\} x := \text{exp} \{P\}} \text{aff}$$

Exemple :

$$\frac{}{\{y = 2\} x := y+5 \{x = 7\}} \text{aff}$$

car $x = 7[x \mapsto y+5] \Leftrightarrow y + 5 = 7 \Leftrightarrow y = 2$

Calcul de Hoare : affectation

Axiome pour l'affectation

$$\frac{}{\{P[x \mapsto exp]\} x := exp \{P\}} \text{aff}$$

Exemple :

$$\frac{}{\{m = 2n\} m := m - n \{m = n\}} \text{aff}$$

car $m = n[m \mapsto m - n] \Leftrightarrow m - n = n \Leftrightarrow m = 2n$

Calcul de Hoare : affectation

Axiome pour l'affectation

$$\frac{}{\{P[x \mapsto exp]\} x := exp \{P\}} \text{aff}$$

Exemple :

$$\frac{}{\{true\} z := 3 \{z = 3\}} \text{aff}$$

car $z = 3[z \mapsto 3] \Leftrightarrow 3 = 3 \Leftrightarrow true$

Calcul du Hoare : séquence

Règle pour la séquence :

$$\frac{\{P\} \text{ ins}_1 \{Q\} \quad \{Q\} \text{ ins}_2 \{R\}}{\{P\} \text{ ins}_1 ; \text{ ins}_2 \{R\}} \text{seq}$$

Apparition d'une **propriété intermédiaire** Q qui soit vraie après ins_1 et telle que R puisse être prouvée à partir de Q après ins_2

Calcul du Hoare : séquence

Règle pour la séquence :

$$\frac{\{P\} \text{ ins}_1 \{Q\} \quad \{Q\} \text{ ins}_2 \{R\}}{\{P\} \text{ ins}_1 ; \text{ ins}_2 \{R\}} \text{seq}$$

Exemple :

$$\frac{\{y = 1\} \text{ x} := \text{y} + 1 \{ ? \} \quad \{ ? \} \text{ z} := \text{x} - 1 \{z = 1\}}{\{y = 1\} \text{ x} := \text{y} + 1 ; \text{ z} := \text{x} - 1 \{z = 1\}} \text{seq}$$

Calcul du Hoare : séquence

Règle pour la séquence :

$$\frac{\{P\} \text{ ins}_1 \{Q\} \quad \{Q\} \text{ ins}_2 \{R\}}{\{P\} \text{ ins}_1 ; \text{ ins}_2 \{R\}} \text{seq}$$

Exemple :

$$\frac{\frac{\{y = 1\} \quad x := y + 1 \quad \{x = 2\}}{\text{aff}} \quad \frac{\{x = 2\} \quad z := x - 1 \quad \{z = 1\}}{\text{aff}}}{\{y = 1\} \quad x := y + 1 \quad ; \quad z := x - 1 \quad \{z = 1\}} \text{seq}$$

Calcul du Hoare : séquence

Règle pour la séquence :

$$\frac{\{P\} \text{ ins}_1 \{Q\} \quad \{Q\} \text{ ins}_2 \{R\}}{\{P\} \text{ ins}_1 ; \text{ ins}_2 \{R\}} \text{seq}$$

Exemple :

$$\frac{\{S = X^{N-P}\} S := S * X \{ ? \} \quad \{ ? \} P := P - 1 \{S = X^{N-P}\}}{\{S = X^{N-P}\} S := S * X ; P := P - 1 \{S = X^{N-P}\}} \text{seq}$$

Calcul du Hoare : séquence

Règle pour la séquence :

$$\frac{\{P\} \text{ ins}_1 \{Q\} \quad \{Q\} \text{ ins}_2 \{R\}}{\{P\} \text{ ins}_1 ; \text{ ins}_2 \{R\}} \text{seq}$$

Exemple :

$$\frac{\frac{}{\{S = X^{N-P}\} \text{ S} := \text{S} * \text{X} \{S = X^{N-P+1}\}}{\text{aff}} \quad \frac{}{\{S = X^{N-P+1}\} \text{ P} := \text{P} - 1 \{S = X^{N-P}\}}{\text{aff}}}{\{S = X^{N-P}\} \text{ S} := \text{S} * \text{X} ; \text{ P} := \text{P} - 1 \{S = X^{N-P}\}} \text{seq}$$

Calcul de Hoare : conséquence

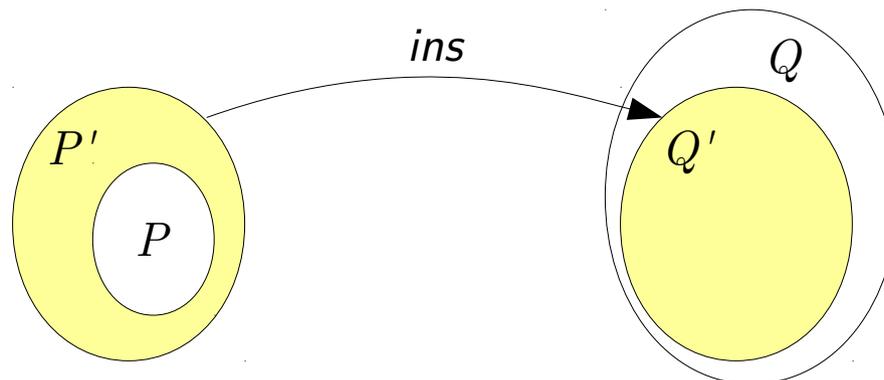
Règle de conséquence :

$$\frac{P \Rightarrow P' \quad \{P'\} \text{ ins } \{Q'\} \quad Q' \Rightarrow Q}{\{P\} \text{ ins } \{Q\}}_{\text{cons}}$$

Possibilité d'affaiblir P ou de renforcer Q pour prouver $\{P\} \text{ ins } \{Q\}$

Si $\{P'\} \text{ ins } \{Q'\}$ est valide, alors il est valide en particulier :

- pour un sous-ensemble des entrées satisfaisant P' , défini par P
- pour un sur-ensemble des sorties satisfaisant Q' , défini par Q



Calcul de Hoare : conséquence

Règle de conséquence :

$$\frac{P \Rightarrow P' \quad \{P'\} \text{ ins } \{Q'\} \quad Q' \Rightarrow Q}{\{P\} \text{ ins } \{Q\}}_{\text{cons}}$$

Possibilité d'affaiblir P ou de renforcer Q pour prouver $\{P\} \text{ ins } \{Q\}$

Exemple :

$$\frac{x \geq 10 \Rightarrow x \geq 0 \quad \{x \geq 0\} \text{ prog } \{y = 1\} \quad y = 1 \Rightarrow y \geq 0}{\{x \geq 10\} \text{ prog } \{y \geq 0\}}_{\text{cons}}$$

Calcul de Hoare : conséquence

Règle de conséquence :

$$\frac{P \Rightarrow P' \quad \{P'\} \text{ ins } \{Q'\} \quad Q' \Rightarrow Q}{\{P\} \text{ ins } \{Q\}} \text{cons}$$

Possibilité d'affaiblir P ou de renforcer Q pour prouver $\{P\} \text{ ins } \{Q\}$

Applications particulières :

$$\frac{P \Rightarrow P' \quad \{P'\} \text{ ins } \{Q\}}{\{P\} \text{ ins } \{Q\}} \text{cons} \qquad \frac{\{P\} \text{ ins } \{Q'\} \quad Q' \Rightarrow Q}{\{P\} \text{ ins } \{Q\}} \text{cons}$$

Calcul de Hoare : IF THEN ELSE

Règle pour le branchement conditionnel

$$\frac{\{P \wedge cond\} ins_1 \{Q\} \quad \{P \wedge \neg cond\} ins_2 \{Q\}}{\{P\} \text{ IF } cond \text{ THEN } ins_1 \text{ ELSE } ins_2 \{Q\}} \text{ if}$$

Pour que Q soit vraie après le branchement, il faut qu'elle soit vraie **quelle que soit la branche exécutée** : preuve divisée en deux sous-cas

Calcul de Hoare : IF THEN ELSE

Règle pour le branchement conditionnel

$$\frac{\{P \wedge cond\} ins_1 \{Q\} \quad \{P \wedge \neg cond\} ins_2 \{Q\}}{\{P\} \text{ IF } cond \text{ THEN } ins_1 \text{ ELSE } ins_2 \{Q\}} \text{ if}$$

Exemple :

$$\frac{\frac{\{-4 \leq x \leq 4 \wedge x \geq 0\} x := x - 2 \{-2 \leq x \leq 2\}}{\text{aff}} \quad \dots}{\{-4 \leq x \leq 4\} \text{ IF } x \geq 0 \text{ THEN } x := x - 2 \text{ ELSE } x := x + 2 \{-2 \leq x \leq 2\}} \text{ if}$$

Calcul de Hoare : IF THEN ELSE

Règle pour le branchement conditionnel

$$\frac{\{P \wedge cond\} ins_1 \{Q\} \quad \{P \wedge \neg cond\} ins_2 \{Q\}}{\{P\} \text{ IF } cond \text{ THEN } ins_1 \text{ ELSE } ins_2 \{Q\}} \text{ if}$$

Exemple :

$$\frac{\dots \quad \frac{\{x < 0\} x := -x \{x \geq 0\}}{\{true\} \text{ IF } x < 0 \text{ THEN } x := -x \text{ ELSE } \text{ SKIP } \{x \geq 0\}} \text{ skip}}{\{true\} \text{ IF } x < 0 \text{ THEN } x := -x \text{ ELSE } \text{ SKIP } \{x \geq 0\}} \text{ if}$$

Preuve d'un programme

Soit le programme suivant :

```
IF x > y
THEN max := x
ELSE max := y
```

Entrées : x, y

Sortie : \max

Donner la spécification de ce programme et démontrer la validité du triplet de Hoare correspondant

Calcul de Hoare : WHILE

Règle pour la boucle

$$\frac{\{P \wedge cond\} \text{ ins } \{P\}}{\{P\} \text{ WHILE } cond \text{ DO } \text{ ins } \{P \wedge \neg cond\}} \text{ while}$$

La propriété P doit être vraie à l'initialisation et à la sortie de la boucle, et doit être préservée à chaque tour de boucle

P : invariant de boucle

Difficulté majeure de la preuve de programme : trouver cet invariant

Calcul de Hoare : WHILE

Règle pour la boucle

$$\frac{\{P \wedge cond\} \text{ ins } \{P\}}{\{P\} \text{ WHILE } cond \text{ DO } \text{ ins } \{P \wedge \neg cond\}} \text{ while}$$

L'invariant de boucle P doit être vrai à l'initialisation et à la sortie de la boucle, et doit être préservé à chaque tour de boucle

Exemple : $P \Leftrightarrow x \geq 0$

$$\frac{\frac{}{\{x > 0\} \text{ x } := \text{ x}-1 \{x \geq 0\}} \text{ aff}}{\{x \geq 0\} \text{ WHILE } \text{ x } > 0 \text{ DO } \text{ x } := \text{ x}-1 \{x = 0\}} \text{ while}$$

Calcul de Hoare : WHILE

Règle pour la boucle

$$\frac{\{P \wedge cond\} ins \{P\}}{\{P\} \text{ WHILE } cond \text{ DO } ins \{P \wedge \neg cond\}} \text{while}$$

L'invariant de boucle P doit être vrai à l'initialisation et à la sortie de la boucle, et doit être préservé à chaque tour de boucle

En général, règle de conséquence nécessaire pour introduire l'invariant

$$\frac{P \Rightarrow Inv \quad \frac{\{Inv \wedge cond\} ins \{Inv\}}{\{Inv\} \text{ WHILE } cond \text{ DO } ins \{Inv \wedge \neg cond\}} \text{while} \quad Inv \wedge \neg cond \Rightarrow Q}{\{P\} \text{ WHILE } cond \text{ DO } ins \{Q\}} \text{cons}$$

Preuve d'un programme

Soit le programme suivant

```
X := 1 ;  
S := 1 ;  
WHILE X ≤ N DO  
    S := S * X ;  
    X := X + 1
```

Entrée : N

Sortie : S

Trouver *Inv* tel que :

- *Inv* soit vrai à l'entrée de la boucle
- *Inv* soit vrai à la sortie de la boucle
- *Inv* soit préservé à chaque tour de boucle